

AD-A045 350

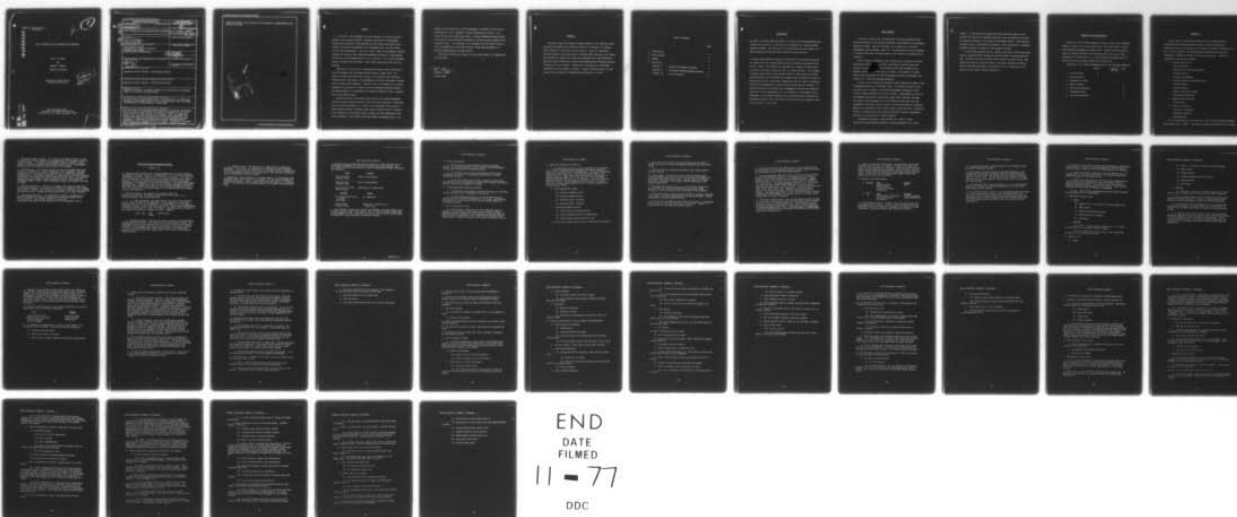
SACRAMENTO AIR LOGISTICS CENTER MCCLELLAN AFB CALIF D--ETC F/6 9/2
USE OF ADVANCED ADP DEVELOPMENT/DESIGN TECHNIQUES.(U)
MAY 77 J E VITTON, A M CARLSON

UNCLASSIFIED

SM-AC/ACD-77-01

NL

1 OF 1
AD
A045 350



END
DATE
FILMED

11 - 77

DDC

REPORT NO. SM-AC/ACD-77-01
31 May 1977

AD A 045350

USE OF ADVANCED ADP DEVELOPMENT/DESIGN TECHNIQUES

JOYCE E. VITTON

AND

ALBERT M. CARLSON

Computer Specialists

Approved for Public Release
Unlimited Distribution



AD No. _____
DDC FILE COPY

Data Automation Branch
Sacramento Air Logistics Center/ACD
McClellan Air Force Base, California 95652

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 14 SM-AC/ACD-77-01	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER 9
4. TITLE (and Subtitle) 6 Use of Advanced ADP Development/Design Techniques	5. TYPE OF REPORT & PERIOD COVERED Case Study-Final Report	
7. AUTHOR(s) 10 Joyce E. Vittone and Albert M. Carlson	6. PERFORMING ORG. REPORT NUMBER	
8. CONTRACT OR GRANT NUMBER(s)	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Data Automation Branch Sacramento Air Logistics Center/ACD McClellan AFB CA 95652	12. REPORT DATE 11 31 May 1977	
11. CONTROLLING OFFICE NAME AND ADDRESS	13. NUMBER OF PAGES 50	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 12 47p.	15. SECURITY CLASS. (of this report)	
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE		
16. DISTRIBUTION STATEMENT (of this Report) Approved for Public Release - Unlimited Distribution		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) Approved for Public Release - Unlimited Distribution		
18. SUPPLEMENTARY NOTES Prepared as the result of students comments on and comparison of techniques taught in two similar training courses.		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Top Down Design; Top Down Documentation; Structured Walk-Thru; Structured Programming; Development Support Library; Team Operations; Data Flow Graph, Transform Analysis; Design Analysis; Pseudo Code.		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This paper deals with changing concepts emerging in the design and implemen- tation of automated data systems and efforts at Sacramento Air Logistics Center (SM-ALC) to teach and use these concepts. After two commercially marketed training courses were given to selected SM-ALC employees, the trainees were required to evaluate the courses and techniques which each promulgated. Based upon the evaluations, a unique development/design methodology has been formulated for use at SM-ALC. Further work on prepara- tion of development standards is also in progress at SM-ALC. A paper		

reflecting results of this project will be prepared in approximately nine months to one year.

✓

ACCESS	
NTIS	DDC
UNANNOUNCED	
JUSTIFICATION	
BY	
DISTRIBUTION/AVAILABILITY CODES	
Dist.	Avail.
SPECIAL	

A

FORWARD

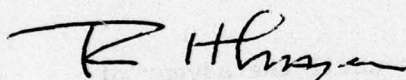
In July 1974, two programmers from the Sacramento Air Logistics Center (SM-ALC), Data Automation Branch, attended an IBM course on "Programming Productivity Techniques" which embodied the new software engineering philosophies of Structured Programming, Chief Programmer Teams, Top Down Design, and Top Down Implementation. Upon their return, these two programmers conducted a number of courses on these advanced programming techniques to approximately one third of the programmer/analyst staff at this ALC. All of these programmer/analyst personnel were encouraged to apply these techniques within their own workload.

There was no serious effort at applying the concepts until the advent of a new development work for Foreign Military Sales in January 1976. As a result of trying to use these new concepts, it was apparent that sufficient training had not been given to the programmers of this new development workload on how to apply these techniques. Because of this, we arranged for an on-base presentation by Yourdon Inc. titled "Structured Design/Programming Workshop." (We made attendance of the Sacramento ALC Advanced Techniques Class a prerequisite to the Yourdon class.)

During the course of the Yourdon Class, many of the student/programmers voiced very strong opinions about the worth of the IBM techniques in comparison with the Yourdon techniques. As a result it was determined that in order to capture these opinions, a detailed survey would be asked of these 14 students at the completion of the Yourdon Class to compare the IBM Programming Productivity Techniques to the Yourdon Structured Design/ Programming system. The

purpose of this survey was to enable management to determine the best mix of technologies to use to implement a software development at SM-ALC. As a result of this survey and further study, a software engineering design system to be used for future software developments at SM-ALC has been prepared and is currently in use. It is evolving, it is changing, and will shortly result in a set of standards for the orderly analysis, design and development of a reliable and maintainable software system.

This report contains the results of this student survey, our findings and our conclusions.



RICHARD H. THAYER

Colonel, USAF

ABSTRACT

This paper deals with changing concepts emerging in the design and implementation of automated data systems and efforts at Sacramento Air Logistics Center (SM-ALC) to teach and use these concepts. After two commercially marketed training courses were given to selected SM-ALC employees, the trainees were required to evaluate the courses and techniques which each promulgated. Based upon these evaluations, a unique development/design methodology has been formulated for use at SM-ALC. Further work on preparation of development standards is also in progress at SM-ALC. A paper reflecting results of this project will be prepared in approximately nine months to one year.

TABLE OF CONTENTS

	PAGE
1. Introduction	1
2. Study Contents	2
3. Summary	4
4. Conclusions	5
Appendix I - Advanced Programming Techniques	7
Appendix II - Structured Design/Programming Workshop	9
Appendix III - Class Evaluations	11

INTRODUCTION

Two types of training (IBM and Yourdon) in the use of new design/documentation techniques were given to a control group of Sacramento Air Logistics Center (SM-ALC) employees. The IBM course was provided first and some limited use of IBM techniques had been tried before Yourdon training was given.

It became readily apparent during the conducting of the Yourdon training that the students had very strong viewpoints about the material presented in both courses. Many of the students were outspoken as to whether the IBM version was better, the Yourdon version was best, or neither was good. In an attempt to capture these ideas for the eventual benefit of other people in the Data Automation Branch, it was decided that students would be required to write an evaluation of the two courses and a comparison of the two types of techniques along with recommendations of which techniques should be used at SM-ALC. It should be noted that the students were programmers of SM-ALC Data Automation Branch and if this organization was to be successful in using modern design techniques, it was vitally important that we understand which techniques our programmers wanted to use. The results of these surveys are contained in and are the reason for this report.

STUDY CONTENTS

In the past several years, the philosophy concerning techniques which should be employed in data systems development and documentation has undergone significant change. Many new techniques are in vogue such as top down programming, chief programmer teams, etc. The application of these techniques is viewed by many data processing leaders as the current and accepted mode of operation.

From a practical standpoint, these techniques may be applied in various combinations depending upon the user's philosophy of operation and needs. Formulas for specific use of these techniques have been packaged and are marketed (taught) to potential users by vendors. Two examples of courses based on these techniques are IBM "Programming Productivity Techniques" and Yourdon, Inc., "Structure Design/Programming".

In July 1974, two SM-ALC Data Automation Branch employees attended an IBM "Programming Productivity Techniques" class. The salient features of this class were used to construct an "Advanced Programming Techniques" course outline (reference Appendix 1) for use at SM-ALC. Approximately one third of the computer specialists at SM-ALC were taught these techniques. Although these people were encouraged to use the techniques within their own work assignments, there was no serious effort to use the techniques as a standard method for a development project until January through June 1976. During this period, it became apparent that computer specialists required supplemental training in the application of these techniques.

Arrangements were made to have Yourdon, Inc. conduct a class, "Structure Design/Programming Workshop" (reference Appendix 2 for course

content). The personnel who attended this class were individuals who had attended the "Advanced Programming Techniques" class and generally were those who were also assigned to the referenced development project. This condition provided a base of understanding and ability to make comparisons of techniques previously learned as compared to those taught by Yourdon.

It soon became apparent that the computer specialists involved had strong opinions concerning which techniques and which methods of applying them were best. A decision was made to capture these opinions for future study. Each student was required to prepare written evaluations, comparing the IBM and Yourdon techniques and methods. Presentation of these evaluations is the purpose of this report (reference Appendix 3).

SUMMARY OF CLASS EVALUATIONS

Nearly all of the students suggested a merge of some of the techniques taught by Yourdon, Inc. for the design effort with those taught in the in-house Advanced Programming Techniques Class. Although there were specific likes and dislikes expressed, the underlying tone is that use of these techniques is a "better way", i.e. no one expressed the thought that our present method for designing systems was the proper way.

Specifically, the students recommended use of the following techniques:

	Yourdon	In-House IBM Tech	Both
a. Data Flow Graph	X		
b. Structure Chart	X		
c. Documentation (HIPO)		X	
d. Modular Concept			X
e. Structured Programming			X
f. Programming Teams			X
g. Structured Walk-Thrus			X

CONCLUSION

As the result of information obtained from our students review of training and the experience gained at SM-ALC, we have concluded that techniques from both the IBM and Yourdon approaches should be combined for our use. We have assembled a development/design methodology which has been designated as SACSOFT (Sacramento ALC Software Techniques). SACSOFT is comprised of the following elements:

1. Software Design Tools:

- Data Flow Graphs
- Hierarchy Plus Input/Process/Output
- Structure Charts
- Structured Programming

2. Structured Software Design Philosophy:

- Top-Down Design
- Top-Down Testing
- Modular (Functional) Design
- Structured Walk-Throughs

3. Organizational Structure:

- Team Concepts

4. Delivery procedures:

- Incremental Development
- Incremental Delivery
- Phases/Versions

We are using SACSOFT in a development project (Allied Recoverable Requirements Computations - ARRCs). This system is being developed for use by foreign

governments under the Foreign Military Sales (FMS) Program. In addition, we are currently preparing development standards around this system. The development standards will merge this system with formal configuration management techniques as well as documentation standards (DOD Manual 4120.17M, "Automated Data System Documentation Standards Manual"). When completed, these standards will be presented to HQ AFLC for consideration for command-wide use.

ADVANCED PROGRAMMING TECHNIQUES

1. This 40 hour class was directed to the introduction of seven software engineering techniques, minimal practical application of each, and, in addition, attempted to give attendees a new philosophical attitude towards the programming atmosphere. Each of these will be addressed independently in the following paragraphs.
2. Top Down Design. This lecture presents the tools and methods to accomplish Top Down Design by using Functional Analysis. It introduces the modular concept, a way to identify and resolve programmer problems at the design level, and use of a simplified Data Flow Chart to achieve the top level in a hierarchical design. Once the top level of design was accomplished, total breakdown of each level was encouraged by functional analysis to achieve the total design.
3. Top Down Documentation. The student was introduced to Hierarchy, Input, Process, Output (HIPO) charts to document the system/program. The Hierarchy is a pictorial representation of the functional design. Each function/module on the Hierarchy is defined in detail on the Input, Process, Output charts which are also graphically illustrated to show the data flow from input to output. The HIPO charts use English statements, not programmer terminology, thus they are readily readable by both the user and the programmer.
4. Structured Walk-Thru. The Structured Walk-Thru is a formalized review of all products produced in a development effort, i.e. design, documentation, coding, job control language, etc. The lecture identifies the basic rules for conducting the Walk-Thru and stresses error detection.
5. Top Down Development. Top Down Development consists of developing (programming) the highest level of code first, i.e. the job control language of a system or the controlling module of a program. The necessary linkage to test the control is provided by the use of "stubs". The stub may consist of a portion of the actual code (such as the read or write), may provide code to assist in debugging (such as printout indicating "Module two entered"), or may simply provide entry and exit points to test module linkage.
6. Structured Programming. This lecture presented the do's and don'ts of COBOL to make a program more readable and maintainable. Examples are: Do use the three basic structures, i.e., sequence, if-then-else, and perform instructions; do follow specific indentation and naming conventions; don't use the "alter" or "go to" verbs.

7. Development Support Library. The concept of the Support Library is that it contains the current versions of all documentation and programs, as well as previous versions. It provides the programmer/ analyst a source to review related or interfacing programs and provides project leaders a chronological history as well as a single source for status of the project.

8. Team Operations. This lecture presents the basic programming team which consists of a Team Chief, a Back-up Programmer, and a Librarian. The Chief performs the critical analysis and coding as well as fulfilling leadership responsibilities. The Back-up Programmer is responsible for test and evaluation in addition to providing total back-up capability to the Chief. The Librarian performs various and sundry clerical functions as well as maintaining the Development Support Library. Other programmer/ analysts are added to the team as needed.

9. Practical Application. Teams of 2-3 students were formed on Monday and a class problem assigned. As each lecture or subject was presented, the students were required to develop that portion of the problem. Walk-thrus were performed on HIPOs and coding. Programs were modularized and use of "stubs" required.

10. Programming Atmosphere. The "Psychology of Computer Programming" by Gerald M. Weinberg was required reading with classroom discussion of the various chapters. This reading in addition to the team participation and walk-thru was intended to lead the programmer to an "e

"STRUCTURED DESIGN/PROGRAMMING WORKSHOP"

Yourdon, Inc.

1. This class was presented with the understanding that the local "Advanced Programming Techniques" class was a prerequisite, i.e., all attendees had been introduced to the basic concepts of the structured technology. As a result of this, the instructor was able to modify the standard class and devote eighty percent of the time to lecture and class application of Structured Design methodology. The remainder of class time was devoted to the use and development of Pseudo Code. Although there was no lecture regarding Programmer Teams and Walk-Thrus, the students were required to work in teams throughout the week to solve the problem, and the various stages of the design process were reviewed by use of the walk-thru.

2. Structured Design. The premise of Structured Design is to develop a detailed Data Flow Graph ("Bubble Chart") and convert it to a Structured Design using Transform Analysis.

a. Data Flow Graph. This chart depicts the flow of the conceptual stream of data through the system/program. As the emphasis is on the data flow, this chart is not to be compared to the widely known flow chart which is procedural oriented. As the data flows from input to output, each transformation of the data is depicted in a circle or "bubble". Transformation is defined as any process which effects a change in the data. Example:

Input Trans	Edit	Edited Trans
	Trans	

b. Transform Analysis. The Data Flow Graph is analyzed for determination of the most abstract point of the afferent and efferent streams of data. The transformation which occurs between these two points is known as the "central transform". The programmer then picks up his design by the central transform to create the structured design. Usually this results in one high level module for each major input and output stream of data as well as the central transform module.

c. Design Analysis. The heuristics for analyzing the strengths and weaknesses of the modules were developed on a step-by-step basis throughout the course. They provide the analyst with the necessary tools to identify and resolve programming problems at the design level; and, they lead the analyst into a strong modular structured design.

3. Pseudo Code. Upon completion of the design effort, the programming teams developed Pseudo Code for each of the modules. The code itself consisted of COBOL like statements, however, when doing Pseudo Code there is no concern of syntax or data division, as the primary object is to identify each processing step.

Class Evaluation, Student A

1. Although there were many similarities between the Yourdon and IBM class on structured programming; such as structured walk-thrus, team concepts, etc., the Yourdon class was more advanced in terms of implementation ease, flexibility, and techniques.

<u>Liked</u>	<u>Reason</u>
Data Flow-Chart (Yourdon Class)	Ease of understanding
Structure Chart (Yourdon Class)	Ease of understanding
Structured Walk-Thrus (Both classes)	Improvement of communication

<u>Disliked</u>	<u>Reason</u>
Hierarchic-Input-Process- Output (IBM Class)	Too cumbersome
Pseudo Coding (Yourdon Class)	Impractical, there must be a better way.

3. The techniques I would use to design and implement a new data system would be taken from the "liked" column above. Also, I would use pseudo coding until a better method is found. Only with time and use can the overall effectiveness and approach of these techniques be properly evaluated.

Class Evaluation, Student B

1. Overall evaluation.

a. The Yourdon Structural Design course was worthwhile; however, the presentation was not in the detail that was presented in the local IBM Structured Programming Course.

b. The Yourdon course presented interesting system design techniques, but perhaps of questionable application to current operational systems.

c. To use the Yourdon system design techniques, local policy and standards would be required for current pilot systems; maintenance systems; and new system design or redesign.

2. Techniques/methods liked and disliked.

a. Impressed with the Transform Analysis Technique, but disliked the use of psuedo code as presented in the course.

b. The HIPO approach presented in the local IBM course has good application for system documentation and customer understanding. Specifically, the HIPO is much better than the pseudo coding for customer acceptance.

3. Set of techniques to use.

a. Assuming that a combination of both techniques would be available, I would explore the possibility of using the Yourdon Transfer Analysis Technique for system design, coupled with the IBM HIPO and modular programming. This would satisfy required system documentation, customers acceptance, and systems processing criteria.

Class Evaluation, Student C

1. IBM (Local Version) vs Yourdon Co.

a. The IBM (Local Version) was well presented by both instructors. Overall techniques were meaningful with several excellent points and a few that could be expanded upon. The total HIPO package should be in more detail for a "real-world" system design. The Yourdon version, I felt, should have been longer for the amount of material that was passed out but not covered. Also, the dollar amount charged for the class was too expensive. In house could have done just as well. The techniques were very similar in concept, with just the names being changed; however, Yourdon techniques did go into greater detail.

b. Techniques/methods liked:

- (1) Hierarchy Diagram (Local)
- (2) Overview Diagram (Local)
- (3) Data-Flow Chart (Yourdon)
- (4) Structure Chart (Yourdon)

c. Techniques/methods disliked:

- (1) Visual Table of Contents (Local)
- (2) Detail Diagram (Local) not enough detail
- (3) Pseudo Codes (Yourdon) not sure, yet.

d. Would like a combination of both as each had its own merits.

Class Evaluation, Student D

1. Both the Yourdon class on Structured Design and the locally taught IBM class on Advanced Programming Techniques were very informative.
2. There are specific techniques imbedded in each method which I like and dislike.
3. From Yourdon, I think the structure chart and bubble chart, along with pseudo code, are very good, and provide an excellent tool to discover any design logic errors. However, these are nothing more than a refinement of the things we have always done in the initial design phase.
4. The local IBM version was also very good with the things that were included, but certain techniques were noticeably absent. Therefore, a fair evaluation of the two methods is not possible.
5. Both methods employ top down principles which I endorse, including team concept, structured walk-throughs, top down development, design, programming, testing and implementation.
6. I do not like the HIPO and visual table of contents as programming tools because of their bulk and break in continuity. However, I believe they are very good documentation tools.

Class Evaluation, Student E

1. Both Yourdon and the local IBM version are very informative and can be applied to any development and/or conversion project utilizing the structured top down team walk-through concept.
2. The data flow graph (Yourdon) gives a better overall/semi-detail picture of a problem as compared to the HIPO concept by actually giving a better visual picture of input-transform-output. HIPO's should be kept at an overview level of visual table of contents of the total problem. The structure chart (Yourdon) can be as detailed as a problem requires down to the coding level with the arrows indicating the data and/or indicator flows. A firm top down structure chart (local IBM version) has a tendency to leave open ends or appear to be unbalanced giving a possible misconception of the problem. The pseudo code concept is very valuable in which a designer/programmer has the ability to discover any errors in logic and/or coding prior to coding for keypunch.
3. The pseudo coding is also a very good documentation media whether it be written in the remarks portion of each program and/or included in another form of formal documentation. The top down structured design and programming influenced by the Yourdon concept employing the data flow graph (bubble chart), structure chart and pseudo code would be my consideration in developing, implementing and documenting any data processing system.

Class Evaluation, Student F

1. Overall Evaluation of Techniques: The IBM (Local) and Yourdon techniques are very good, but Yourdon's concepts and "tools" are easier to learn and better to work with. Both use structure charts, and structured programs; both emphasize scope of control, scope of effect, and afferent and efferent data flows with central transform areas. Yourdon also uses data flow graphs (bubble charts) and pseudo code, but does not use HIPO's.

2. Specific Likes and Dislikes:

a. <u>Yourdon</u>	<u>Like</u>	<u>Dislike</u>
	Data Flow Graph	None
	(Bubble Chart)	
	Structured Diagram	
	Pseudo Code	
b. <u>IBM</u>	<u>Like</u>	<u>Dislike</u>
	Visual Table of Contents,	Overview Diagram
	including Extended	Detail HIPO Charts
	Description	

3. Recommended Techniques: Yourdon, with structured design, data flow graphs (bubble charts), structure charts, pseudo code, chief programmer teams, structured walk-throughs in all phases, top down development, coding and testing, and a librarian function.

Class Evaluation, Student G

1. I attended the vendor supplied version of both IBM and Yourdon Structured Programming Classes.
2. Both classes were well taught by qualified instructors. The classes were very similar in content, the main difference being Yourdon's Data Flow Chart and Yourdon does not require a hierarchy in the structure chart. The Yourdon instructor allowed a lot more time for work on class problems. Both classes used a team approach for class problems.
3. The techniques that I especially liked are: the Transformation Analysis by Yourdon, Step Wise Refinement by IBM, and the Pseudo Codes taught by both classes.
4. The techniques that I would use for System Design and Development are: Data Transformation Analysis with the "Bubble Chart" and structure chart, Top Down Testing with a predefined test plan, the Development Library Support Concept, small and frequent walk-thrus, the use of Pseudo Codes, and IPOs only for low level functional descriptions.

Class Evaluation, Student H

1. The Yourdon course deals primarily with "Top-Down" or "Structured" system design on a basis of Data Flow Diagrams or "Bubble Charts"; Structure Charts; Transform and Transaction Analysis as to Function.

a. Much of the above is similar to, but not identical to, the IBM approach. The primary difference is that Yourdon eschews the "HIPO" philosophy and technique of IBM.

b. Both of the techniques are superior to the currently prevalent techniques and philosophies in use at McClellan, with the possible exception of ARRCs. Both emphasize the team concept for design, programming and maintenance of the systems. Also, they are both aimed at the concept of "Egoless" programming.

c. Also - IBM stressed programming more while Yourdon stressed system design prior to "programming".

2. IBM (Local Version) lists:

a. Liked

- (1) Overall idea of structured or top-down design, etc.
- (2) Team concept.
- (3) "Egoless" programming concept.
- (4) Modularization of programs.
- (5) Walk-Thrus.

b. Disliked

(1) "HIPO" - visual table of contents, etc. - it seems too superfluous to a functional overall concept.

(2) Not enough time to do justice to class objectives including use of and writing of Stubs.

3. Yourdon lists:

a. Liked

Class Evaluation, Student H, continued.....

- (1) Again - the overall concept of structured design.
- (2) Team concept.
- (3) "Bubble Charts".
- (4) Transform analysis structure charts.
- (5) Modularization.
- (6) Walk-Thrus.

b. Disliked

(1) Not much - other than occasional break-down of communication due to differences of the two versions (IBM vs Yourdon).

4. If I were to design a system I would probably utilize the techniques from Yourdon primarily. However, it would be practically identical with the IBM technique except for the HIPO philosophy.

a. I particularly like the "Bubble Chart" to Structure Chart thru Transform Analysis to Program Modules or System Modules; in opposition to the HIPO visual table of contents, overview approach of IBM.

b. I like the team concept as well as the idea of doing the design and analysis on the front end as to time requirements instead of a great emphasis on "get it coded and on the machine - we'll correct and de-bug later". This latter philosophy does away with in-depth analysis and design necessary for good system design and programming.

Class Evaluation, Student I

1. Both the IBM and YOURDON design techniques seemed impressive in the classroom, but I had difficulty applying IBM techniques to a D049 Subsystem design and I could not successfully apply YOURDON's techniques to the classroom problem.
2. I like the idea of a standardized top down method of design, but I did not really like (or didn't understand) any of the specific techniques presented - they all seemed too vague.
3. If I had enough time to learn as the design project progressed, I think that I would review in depth the materials from the YOURDON class and attempt a design using their techniques. If time was a critical factor, I would probably use the old methods most familiar to me.

Class Evaluation, Student J

Both the Yourdon and IBM techniques are excellent. Specific techniques that I liked about each of the two approaches were top down design, top down development, top down documentation, structured programming, development support library, team concept, and structured walk-throughs. I would use all of these techniques to design and implement a data system.

Class Evaluation, Student K

1. Generally, both the IBM and the Yourdon classes were different, but similar. Techniques of arriving at the same result: Structured Design. Each placed different emphasis on certain stages of the development. To say one or the other is better or worse is a matter of personal preference. Having studied the IBM techniques and having had to apply them, I prefer the IBM over the Yourdon, which I have only studied. An actual application of the Yourdon may change my opinion.

2. Below is listed the steps in each course development, an asterisk marks those I feel are the most beneficial.

IBM

Structured Data Flow
*Visual Table of Contents
*Overview Diagrams
Detail Diagrams

YOURDON

*Data Flow Graph
Structure Chart
*Pseudo Code

3. In designing and implementing a system, I would prefer a combination of the two methods. It would include the following:

- a. Yourdon's Data Flow Graph
- b. IBM's Visual Table of Contents
- c. IBM's Overview Diagram combined with Yourdon's Pseudo Code

Class Evaluation, Student L

1. Evaluation of the Yourdon and IBM (local version) techniques follows:

a. IBM (local version) Technique. This class introduces the concept of seven techniques. They are top down (TD) design which includes functional analysis, HIPOs for documentation, TD development using stubs, structured programming, walk-throughs, programmer teams, and the development library. A simple class problem is developed through all the stages. Through the experience of teaching the class several times, I recognized the tool for implementing all the techniques was using functional analysis in design. However, the materials provided us by IBM did not carry the concept far enough and there was not enough emphasis on the design in the local class.

b. Yourdon Technique. The class, though almost entirely devoted to design, introduced only one new technique and that was the use of pseudo code. It also included practical application in working in teams and using walk-throughs. The design concept is basically the same as taught in the local version with the big exception that it is exploded into more detail and was carried several steps beyond the local version. The Yourdon class provided me with the missing links in the design effort.

2. As the techniques are so similar in my eyes, I can only see a merge of the two. Specifically, I liked the Yourdon technique to demonstrate flow of data on the design chart. I preferred the HIPO as documentation, because it shows input/output as well as processing. I can see the use of pseudo code in the processing block of the HIPO. All discussion and material provided regarding the other techniques were very similar.

3. If I were to design and implement a data system, I would use all of the IBM techniques as modified by application of the Yourdon techniques described in paragraph 2 above.

Class Evaluation, Student M

1. The IBM and Yourdon classes both present practical approaches to system design.

a. The IBM class was interesting and well taught. The work groups were able to do the class problem with moderate difficulty. Not all groups completed the problem because of poor turnaround on the 370. The instructors did a good job but neither one had any experience using the system outside the classroom.

b. The Yourdon instructor did a very good job. He is a consultant who has written a major system using the techniques that he teaches. The work groups all had some problems but everybody seemed to get the idea by the end of the class. It was impossible to do all the reading.

2. IBM and Yourdon classes were very similar in some areas and quite different in others. The differences were primarily a matter of emphasis.

a. The IBM data flow chart is too brief to be useful. The Yourdon data flow chart is quite detailed. This detail makes it a useful tool.

b. The Yourdon structure chart is more flexible than the IBM hierarchy diagram. Data flow is indicated by arrows in the Yourdon structure chart. The IBM hierarchy diagram looks neater but one must look at the lower level diagrams to see data flow.

c. IBM describes the detail in a HIPO chart. Yourdon describes the detail with pseudo code. The pseudo code is easier to use and it is easier to code from. Pseudo code is also more apt to show logic errors because it is closer to the actual program.

d. IBM strongly emphasizes the structured walk-through. It is only mentioned by Yourdon. The walk-through is beneficial.

3. The following is a summary of the major steps that could be used in designing a data system.

a. Make a data flow diagram showing the data flow of the proposed system. This diagram should be quite detailed.

b. Using the data flow diagram make a structure chart of the system. Considerable effort should go into this step.

Class Evaluation, Student M, continued.....

- c. Pseudo code each block of the program. Minor changes to the structure chart may be necessary at this time.
- d. Code the modules from the pseudo code.
- e. Test the system.
- f. Structured walk-through should be used after each phase.

Class Evaluation, Student N

1. My evaluation of both courses on structured programming is favorable.
2. They were two different courses with the same basic goal of having a better way to design, modify, and maintain a system.
3. The instructors had good knowledge of the subjects they were teaching.
4. The class problem:
 - a. Was useful in helping us implement what we were supposed to learn.
 - b. Everyone was involved.
5. Both classes had adequate books and handouts; the Yourdon course had more material.
6. The Yourdon course gave a broader view with more techniques and ideas.
7. Attending the in-house IBM course first, provided a foundation for the Yourdon course.
8. Some techniques I liked.
 - a. Structured programming. "Each piece of the system corresponds to exactly one small, well-defined piece of the problem and each relationship between system pieces corresponds only to a relationship between pieces of the problem."
 - b. Modular Programming.
 - (1) Easier to write and test separately.
 - (2) Easier to debug, control and maintain.
 - c. Bubble Charts or Data Flow Graphs.
 - (1) Shows the flow of data.
 - (2) The Transform Analysis Strategy makes it easier to switch to a structure chart and produce a hierarchy chart of the system.

Class Evaluation, Student N, continued.....

d. Pseudo Language.

- (1) Could be a useful tool before coding.
- (2) Could discover possible logic problems that might have been overlooked.

e. Team Work.

- (1) Egoless programming.
- (2) Exchange of ideas.
- (3) Work may be accomplished in sections by small productive groups.

9. Essential Techniques for System Design and Implementation.

a. Top-Down Structured Design.

- (1) Bubble Charts.
- (2) Transform Analysis Strategy.
- (3) Structured Chart of the Hierarchy (very important, shows control).
- (4) Put interrelated items in the same area of the system.

b. Top-Down Coding. Using dummy routines where necessary.

c. Top-Down Documentation.

- (1) HIPO Diagrams (very important, shows what the module does).
 - (a) Avoid the fat arrows.
 - (b) Break the program into smaller pieces until each module is reasonably small.

(2) Pseudo Language.

d. Structured Walk-Throughs.

Class Evaluation, Student N, continued.....

(1) Within your group before you progress to outside your group (user).

(2) Avoid redesigning the entire program unless you are at the beginning.

(3) Ensure your techniques are correct.

(4) Use real input data if possible and produce real output.

e. Team Concept.

(1) Informal leadership.

(2) If something is wrong with the program, blame the entire team not the individual.

(3) When debugging after hours, call the whole team, not just one individual.

f. Be flexible.

10. Other techniques which are useful.

a. Avoid non-mnemonic data-names.

b. Every data-name and paragraph (maybe) should be meaningful or understandable.

c. Paragraphs should be numbered.

d. Keep paragraph names in numerical order.

e. Perform only one paragraph at a time unless it has an exit; perform only through the exit paragraph.

f. Insert comments only if they are meaningful and help to clarify.

g. Comment the program while coding not afterwards.

h. If the statement requires more than one line:

(1) Do not hyphenate in the middle of a variable name or a constant.

Class Evaluation, Student N, continued.....

(2) Make the break in a readable fashion.

i. Avoid unnecessary complex expressions.

j. Avoid jumping in and out of loops.

k. Make subordinate modules complete and sufficient to implement the module's function.

l. There is no definite limit on the size of a module, but try to keep it small.

m. Avoid allowing one module to fall into another.

n. All "IF and ELSE" statements should be indented.

o. Within the ELSE portion, indent the "IF and ELSE" statement.

p. Avoid GO TOs unless:

(1) To go to an exit.

(2) Go to depending and returning when you have a large number of different input records.

Class Evaluation, Student 0

1. My evaluation of both the IBM (local version) and the Yourdon Course in structured programming is favorable. The instructors had good knowledge of the subjects and used a class problem which made the courses meaningful.

2. Listed below are the specific techniques I like regarding the Yourdon and local IBM courses.

a. Team work allows for:

(1) Exchange and reaffirmation of ideas.

(2) Work accomplished in increments (phases) which show results early in the development stage of the system.

(3) Walk-throughs of design and coding which should minimize testing time.

(4) Testing of system by the user who knows what he expects as output.

b. Modularity makes system easier to debug and maintain.

c. Data flow graphs were emphasized much more in the Yourdon Course. This technique gives an overall view of the stream of data elements. It also allows for a transition to a structure chart which shows the hierarchy and organization of system functions.

d. Pseudo language was presented in the Yourdon Course only. Its use was not well defined. It could be a very useful tool for pre-coding walk-throughs if about 6 to 8 common verbs were defined.

3. The following is a list of the techniques I think are essential for system design and implementation.

a. Top down structured design.

(1) Data flow graphs.

(2) Structured charts - tea cup approach of development. Design, code and test using real input and producing real input. Execute outer branches of hierarchy of structured charts and work inward.

Class Evaluation, Student 0, continued.....

b. Top down documentation.

(1) HIPOs at first level breakout of structured chart.

(2) Pseudo coding for detail using pre-defined verbs and writing structure.

c. Team concept with a chief programmer.

d. Structured walk-throughs using the data flow graphs for walk-throughs with user.

Class Evaluation, Student P

1. The Yourdon Class ideas and concepts are exceptionally good.
2. Benefits to be derived by participants could be improved by:
 - a. Instructor presentation of two small problems. The presentation should be a complete presentation, as:
 - (1) Bubble Chart
 - (2) Functional Chart
 - (3) Pseudo Code
 - (4) COBOL Statements
 - b. By instructor presentation of two complete problems, the class benefits by observing correct preparation of each technique. Then the student is reinforced by seeing the second problem solution. He is now prepared to tackle a team problem and knows exactly how to go about it. This method would cut time spent in team problem solution by at least one-half and would not add appreciably to time spent on class presentation since the instructor is already adept at solving problems using these techniques.
3. The techniques I liked were:
 - a. Proper preparation of the bubble chart allows you to see the exact functional chart.
 - b. The analogy to management structures.
 - c. Examples of cohesion.
4. Techniques I did not like:
 - a. It seems the coding of the problem using pseudo code was an extra unnecessary process. I did not see the usefulness of this step. I may be able to code using pseudo code as it certainly is not difficult but whether I can then apply COBOL to it is another concern.
5. The local version of the IBM technique is also a good class. My feeling is much the same as my comments about the Yourdon class. Specifically:

Class Evaluation, Student P, continued.....

a. There is no instructor presentation in solution form of the techniques presented in class. Rather pieces of information are thrown to the class and then a class problem presented. Two small problems completely presented and followed through in proper format by the instructor, answering questions and explaining techniques, would be far more valuable to class attendees.

b. I personally see no requirement for the "down with management" sessions that are held. There is enough criticism without encouraging more. If some good is brought about by this, it has escaped me. For me, one of these sessions would be adequate - it is not needed everyday.

6. Techniques I liked were:

a. The top down approach to design and programming.

b. The use of stubs in coding.

c. When you have completed you program using the described techniques, your documentation is completed.

d. Not having to detail flow chart.

e. The structured walk-throughs are very worthwhile. These definitely detect problem areas early in the design phase. The exchange of ideas this allows.

f. Orderly test development.

g. The use of the control module.

7. Techniques I did not like:

a. The lack of problem presentation in class discussed in paragraph 5a results in instructor manipulation of the teams during solution of the class assigned problem.

b. The daily sessions of criticizing management. This can be accomplished in another outlet.

8. To design and implement a data system I would:

a. Obtain two computer specialists who have extensive knowledge of the area to be designed. These people should be agreeable and pliable.

Class Evaluation, Student P, continued.....

- b. Have these two individuals prepare the data flow and the functional chart (in detail).
- c. Obtain Mission agreement on this portion.
- d. Have a series of group meetings developed by the two computer specialists explaining to the selected programmers the overall design of the system.
 - (1) explain the inputs
 - (2) explain the outputs
 - (3) explain the process as envisioned by the specialists
- e. Define the data elements.
- f. If needed, pick up more computer specialists.
- g. Assign project processes to teams for development.
- h. Apply the structured walk-through technique.
- i. Apply top down programming. Use HIPO charts.
- j. Have group meetings to bring everyone up to date on project development.

Class Evaluation, Student Q

1. Overall evaluation of Yourdon and IBM (Local Version) Techniques:

a. The IBM course (local version) was a good course to introduce some of the basic ideas of top down design and structured code. It also briefly introduced the important concepts of binding and coupling, and how they relate to the design of a "good module" in a program. Overall, the IBM course was a course which introduced concepts, but left me with very few techniques to implement them.

b. The Yourdon course reviewed the basic concepts covered in the IBM course, but went on to provide some useful techniques in application. Some of these techniques I have found useful on the job. I think this is an exceptional opportunity for our system analysts and programmers to acquire new skills and knowledge of some of the latest concepts and techniques in the field of systems design and application programming. These new skills would allow us to greatly improve the design and operation of future systems. The Yourdon course has provided us with a wealth of new ideas and some promising application tools. If we are to ever begin to effectively utilize the advances in new hardware, we must take advantage of these new developments in application techniques.

2. List of specific techniques/methods that I liked and disliked about each of the two approaches:

a. List of techniques or methods I liked about the Yourdon class:

(1) Transform analysis using the data flow graph to get a top down design.

(2) Transaction analysis when transform analysis is not practical.

(3) Structure chart used to modularize the design obtained above. I especially liked the systematic methods used to convert the data flow graph to a structure chart.

(4) Pseudo code used as a method of generating COBOL code consistent with the structure chart.

(5) Structured COBOL coding.

Class Evaluation, Student Q, continued.....

(6) I liked the idea of getting higher level modules running for an entire system i.e. getting some kind of data flowing through the system as soon as possible after the design is established using skeleton programs and modules when necessary in incomplete parts of the system.

b. List of techniques or methods I liked about the IBM class:

(1) Structured coding:

- (a) Set of rules for indentation
- (b) Use of IF ELSE
- (c) Use of PERFORM UNTIL
- (d) Rule for not using "GO TO" statements except to an exit, and limiting this where practical.
- (e) One statement per line.

(2) Use of stubs for testing incomplete programs.

(3) One entry and one exit for a module.

c. List of techniques or methods I disliked about the Yourdon class:

(1) Pseudo coding when defined as a structured language. I view pseudo code as a working tool to transform a structure chart to a coded program. As such, it becomes a "roughed out" version of the COBOL and need not follow definite syntax rules. If a programmer can write COBOL from his/her own pseudo code, it is effective. I assume here that the programmer is not writing the pseudo code for others.

(2) Pseudo coding when it is required in all cases before coding. I think it is possible to get "hung up" in pseudo code to the point where it loses its effectiveness. Generally pseudo code is an aid to COBOL coding, but there are cases when it is unnecessary and a duplication of effort. Some coding can be done directly from a structure chart.

d. List of techniques or methods I disliked about the IBM class:

Class Evaluation, Student Q, continued.....

(1) Top Down approach to getting a top down design, i.e. the method of first blocking out the problem in general terms or modules and then breaking them down into more detail until reaching the level to code. The top down approach is too intuitive and tends to be difficult to apply. The successful application of this approach depends more on the experience and expertise of the analyst or programmer than it does on the method itself. As a working tool to assist a person in designing a well structured, top down program with functionally bound, loosely coupled modules, this approach is very limited.

(2) HIPOS - this whole concept is so cumbersome that it is practically impossible to design and maintain a changeable system when using it. As a documentation tool, it suffers from the same problem that flowcharts do, i.e. it is awkward to update and in the working environment will usually not reflect the operating system.

3. Set of techniques to design and implement a data system:

a. Confirm inputs and outputs of system.

(1) Write up specifications for customer outputs identifying areas of possible/probable change. This includes the formatting of all reports.

(2) Write up specifications for customer inputs. This includes the format of both data input and report requests. Some of the inputs, necessary to generate the required reports, may not be obvious at this point.

b. Make a data flow diagram, (bubble chart), of the entire system using the techniques of transform analysis and transaction analysis. (Yourdon Chapter 10.2)

(1) This diagram should contain enough detail to allow the resulting structure chart to be packaged into programs. It should contain enough detail to write basic program specifications. Too much detail is better than too little.

(2) Previously unidentified input requirements will become obvious during this step. Coordinate changes in input requirements with the customer.

(3) Limitations in output products will become obvious during this step. Some output products may be impractical to provide to the customer. Discuss and coordinate changes.

Student Evaluation, Student Q, continued.....

(4) Identify possible/probable areas of change and design accordingly.

c. Make a structure chart for the entire system. (Yourdon Chapters 9 thru 14)

(1) Identify major afferent (input) streams.

(2) Identify major efferent (output) streams.

(3) Identify central transform bubble(s).

(4) Make a top level structure chart.

(5) Factor central transform into subordinates. This may be the most difficult step. It is made simpler if, during the design of the data flow diagram, the number of bubbles in the central transform is minimized, i.e. if the central transform contains mostly control functions and very few transform functions. The central transform on the CYBER, for example can be thought of as the JCL.

(6) Factor afferent modules into subordinates.

(7) Factor efferent modules into subordinates.

(8) Determine packaging, grouping modules into programs (Yourdon Chapter 14)

(9) Identify program Inputs and Outputs.

(10) Assign data names and formats to program inputs and outputs.

(11) Write basic program specifications.

d. Make a data flow diagram for each program using the techniques of transform and transaction analysis.

(1) Detail should be at a level that allows the resulting structure chart to be packaged into program modules. (In many cases, program modules will represent paragraphs in the finished program.)

(2) Previously unidentified input and output data will become visible during this step. Coordinate changes with systems

Student Evaluation, Student Q, continued.....

(3) Identify areas of possible/probable change and design accordingly.

e. Make a structure chart for each program. (Yourdon Chapters 9-14)

(1) Follow steps 1-7 in para 3c above, using the program data flow diagram developed in para 3d. The central transform will in this case correspond to a control module or paragraph in the program.

(2) Identify data flow, control flow, and transforms with short, meaningful names. (Some standardization may be appropriate.)

f. Write pseudo code from the structure chart.

(1) Use names on the chart for paragraph names, data names, flags, etc.

(2) Use pseudo code as a means of "roughing out" the COBOL logic. Abbreviate when COBOL code is obvious.

g. Write coding from pseudo code.

(1) Use rules for structured code.

(2) Use names from pseudo code.

h. Compile and test in stages.

(1) Code top levels of all programs in system.

(2) Use stubs and skeleton program, and "dummy data" where necessary.

(3) Start system testing with top levels.

(4) As programmers develop more code, insert new versions into the system.

(5) Top level testing should start as soon as possible after the completion of the program structure charts (para e).

i. Documentation for the system would be updated as changes were made and could consist of the following:

Class Evaluation, Student Q, continued.....

- (1) System level structure chart (para c).**
- (2) Description of record layouts for files phased between programs.**
- (3) Program specifications (para c(11)).**
- (4) Program structure charts (para e).**
- (5) COBOL program listings (para g, h).**
- (6) Operations instructions.**
- (7) Customer user manual.**